Exploring LLMs for Generating Erroneous Examples in CS1

Yuxuan Chen
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
yuxuan19@illinois.edu

Chenyan Zhao
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
chenyan4@illinois.edu

Kangyu Feng
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
kangyuf2@illinois.edu

Junyu Zhang
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
junyuz6@illinois.edu

Vedan Malhotra
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
vedanm2@illinois.edu

Mariana Silva
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
mfsilva@illinois.edu

Abstract

Erroneous examples are structured problem examples which incorporate deliberate errors that students can identify and correct to reinforce their understanding. Erroneous examples have been shown to be beneficial for student learning in various domains. However, designing effective erroneous examples requires careful planning and a deep understanding of common student misconceptions, posing a time burden on educators. In this work, we introduce a framework that leverages Large Language Models (LLMs) to automatically generate erroneous programming examples for use in introductory computer science education. We systematically evaluate the performance of various LLMs on the generation task, and find that LLMs are generally capable of generating meaningful erroneous examples along with accurate explanations. We present our preliminary findings and outline next steps for this study and future research.

ACM Reference Format:
Yuxuan Chen, Chenyan Zhao, Kangyu Feng, Junyu Zhang, Vedan Malhotra, and Mariana Silva. 2026. Exploring LLMs for Generating Erroneous Examples in CS1. In *Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.2 (SIGCSE TS 2026), February 18–21, 2026, St. Louis, MO, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3770761.3777210

1 Introduction

In computer science (CS) education, instructors commonly use worked examples—complete, correct solutions presented step by step—to guide students in solving questions [3]. Worked examples are effective for modeling problem-solving processes, but they may not sufficiently prepare students to diagnose and recover from errors in their own work. In contrast, erroneous examples are structured problems that contain intentional mistakes and are designed to guide students to identify, explain, and correct those mistakes [8]. This reflective activity encourages deeper cognitive engagement and supports longer-term retention [6]. Erroneous examples have been shown to be effective across a range of educational domains [1, 3]. However, designing high-quality erroneous examples requires careful planning and a deep understanding of the numerous misconceptions students may hold, making the process time-consuming and difficult to scale for instructors.

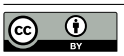
Recent advances in Large Language Models (LLMs) open up the possibility of automating the generation of erroneous examples. LLMs have been increasingly used in educational settings to generate instructional content [5], provide feedback and suggestions to learners [4], and support autograding tasks [9]. These capabilities position LLMs as promising tools for automating tasks that have traditionally been time-consuming for instructors. However, there is limited research on the use of LLMs specifically for generating pedagogically meaningful erroneous examples. It remains unclear whether LLMs can reliably produce realistic and instructive errors that align with common student misconceptions, especially in introductory CS (CS1) contexts.

To investigate this gap, we propose an algorithmic framework that uses LLMs to generate erroneous code examples and associated explanations for CS1 topics. We evaluate outputs under two prompting conditions: zero-shot, which uses no examples, and two-shot, which includes two example and explanation pairs, to assess in-context effects. The results provide initial evidence on the accuracy and comprehensibility of LLM-generated erroneous examples and lay a foundation for future research.

2 Framework and Methodology

Our framework pipeline ingests two primary inputs: (1) task specifications, which may be either a high-level problem description or a correct Python code snippet; and (2) pedagogical schema, comprising a taxonomy of foundational CS1 topics alongside a curated catalog of student misconceptions. These inputs are merged into a prompt that embeds topic-specific buggy exemplars paired with diagnostic explanations. We then dispatch the prompt to one or more state-of-the-art LLMs, which return syntactically valid code snippets exhibiting the targeted misconceptions and natural-language rationales diagnosing each introduced misconception.

For our initial investigation, we evaluated LLM-generated erroneous examples using prompts that specified a variety of programming topics, and restricted outputs to Python language. We adapted a rubric proposed by Scaria et al. [7] for assessing the quality of LLM-generated educational questions, with adjustments to



SIGCSE TS 2026, St. Louis, MO, USA

ACM ISBN 979-8-4007-2255-4/2026/02
https://doi.org/10.1145/3770761.3777210

Rubric item	Description
Python Syntax	Output erroneous example is Python code
Erroneous	Output erroneous example contains an error based on common CS1-level misconceptions
Topic Matching	Output erroneous example matches the provided CS1 programming topic
Explanation Understandable	Output explanation text is grammatically correct and logically coherent
Explanation Matching Erroneous Example	Output explanation text accurately describes and aligns with the output erroneous example

Table 1: Rubric for LLM-generated erroneous example-explanation pairs. Each item is evaluated on a binary scale (True/False).

LLM model	Erroneous	Exp Underst.	Exp Match.
gpt-4o	87.14%	100%	89.29%
gemini-1.5-pro	85.17%	89.29%	85.69%
deepseek-v3	85.00%	97.86%	90.71%
qwen-2.5-72b-instr.	68.57%	97.14%	91.43%

Table 2: Performance of erroneous example–explanation pairs on three rubric items across LLMs (zero-shot).

account for the unique characteristics of erroneous code examples and associated explanations (Table 1).

We identified 14 representative CS1 topics from a published CS1 curriculum [2]. In the zero-shot baseline, we generated five erroneous example-explanation pairs per topic with four models (GPT-4o, Gemini-1.5-Pro, DeepSeek-V3, Qwen-2.5-72B-Instruct), yielding 280 pairs. For the two-shot condition, we narrowed down the CS1 topic set to 7 topics after merging overlapping CS1 topics (e.g., combining "Class definition" and "Class inheritance" into "Classes"). We generated five pairs per topic with Qwen-2.5-72B-Instruct, yielding 35 pairs. Each output was generated via the API and independently evaluated by two graduate students with extensive CS1 background. Inter-rater reliability was computed as percent agreement and exceeded 80% across all rubric items.

3 Results

Raters independently assessed whether each LLM output satisfied each rubric item (True/False). We define the performance metric as the "True" rate averaged across two raters per rubric item and LLM model. In zero-shot condition, three out of four models scored 100% on "Python Syntax" and reached at least 85% on "Erroneous", suggesting a strong ability to generate code examples that contain common CS1 misconceptions. The associated explanations were generally clear and well-aligned, with "Explanation Understandable" from 89.29% to 100% and "Explanation Matching" from 85.69% to 91.43%. GPT-4o led most performance metrics, while Qwen-2.5-72B-Instruct trailed on "Python Syntax" (55.71%) and "Erroneous" (68.57%). Table 2 reports zero-shot performance across LLMs on three of the five rubric items. The results suggest that within the scope of CS1 topics, modern LLMs are capable of generating "accurate" erroneous examples and associated explanations that are clear and well-aligned, even in a zero-shot setting.

To assess the effect of in-context examples, we performed a two-shot evaluation on Qwen-2.5-72B-Instruct, given its lowest zero-shot performance on "Python Syntax" and "Erroneous". Each two-shot prompt included two examples drawn from zero-shot outputs with perfect inter-rater agreement and high-quality ratings for both the erroneous code and its explanation. The performance of Qwen-2.5-72b-Instruct improved substantially on both "Python Syntax" and "Erroneous", increasing from 55.71% to 100% and from 68.57% to 88.57%, respectively. This suggests that two-shot prompting enabled the model to generate code examples that were syntactically correct and intentionally erroneous.

4 Future Work and Conclusion

This paper presents a framework for using LLMs to automatically generate erroneous code examples for CS1 education. We found that modern LLMs are capable of producing valid erroneous examples and corresponding explanations, though their performance varies by model and prompt design. These results show the potential of LLMs as scalable tools to generate pedagogically meaningful erroneous examples for instructors.

Future work includes implementing the framework into a usable tool, conducting instructor user studies to refine the generation workflow, expanding coverage across multiple programming languages, and deploying the tool in classrooms to study how students interact with and learn from LLM-generated erroneous code.

References

[1] Xingliang Chen, Antonija Mitrovic, and Moffat Mathews. 2020. Learning From Worked Examples, Erroneous Examples, and Problem Solving: Toward Adaptive Selection of Learning Activities. *IEEE Transactions on Learning Technologies* 13, 1 (2020), 135–149. doi:10.1109/TLT.2019.2896080

[2] Yuxuan Chen, Chenyan Zhao, Kangyu Feng, Mattox Alan Beckman, and Mariana Silva. 2025. A Complete Redesign of CS1 for Engineering Students. In *2025 ASEE Annual Conference & Exposition*. ASEE Conferences, Montreal, Quebec, Canada. https://peer.asee.org/55349.

[3] Cornelia S. Große and Alexander Renkl. 2007. Finding and fixing errors in worked examples: Can this foster learning outcomes? *Learning and Instruction* 17, 6 (2007), 612–634. doi:10.1016/j.learninstruc.2007.09.008

[4] Mohammed Hassan, Yuxuan Chen, Paul Denny, and Craig Zilles. 2025. On Teaching Novices Computational Thinking by Utilizing Large Language Models Within Assessments. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 471–477. doi:10.1145/3641554.3701906

[5] Breanna Jury, Angela Lorusso, Juho Leinonen, Paul Denny, and Andrew Luxton-Reilly. 2024. Evaluating LLM-generated Worked Examples in an Introductory Programming Course. In *Proceedings of the 26th Australasian Computing Education Conference* (Sydney, NSW, Australia) *(ACE '24)*. Association for Computing Machinery, New York, NY, USA, 77–86. doi:10.1145/3636243.3636252

[6] Bruce M. McLaren, Deanne M. Adams, and Richard E. Mayer. 2015. Delayed Learning Effects with Erroneous Examples: a Study of Learning Decimals with a Web-Based Tutor. *International Journal of Artificial Intelligence in Education* 25, 4 (2015), 520–542. doi:10.1007/s40593-015-0064-x

[7] Nicy Scaria, Suma Dharani Chenna, and Deepak Subramani. 2024. Automated Educational Question Generation at Different Bloom's Skill Levels Using Large Language Models: Strategies and Evaluation. In *Artificial Intelligence in Education*. Springer Nature Switzerland, Cham, 165–179.

[8] Dimitra Tsovaltzi, Erica Melis, Bruce M. McLaren, Ann-Kristin Meyer, Michael Dietrich, and George Goguadze. 2010. Learning from Erroneous Examples: When and How Do Students Benefit from Them?. In *Sustaining TEL: From Innovation to Learning and Practice*. Springer Berlin Heidelberg, Berlin, Heidelberg, 357–373.

[9] Chenyan Zhao, Mariana Silva, and Seth Poulsen. 2025. Language Models are Few-Shot Graders. In *Artificial Intelligence in Education*. Springer Nature Switzerland, Cham, 3–16.